

# INTELLEC<sup>®</sup> SERIES III MICROCOMPUTER DEVELOPMENT SYSTEM POCKET REFERENCE

Order No. 121610-002

intel<sup>®</sup>

*Bruce Oakes*

# TABLE OF CONTENTS

	PAGE
Conventions .....	1
Device Filename Format .....	1
Control Characters .....	2
ISIS-II Console Commands .....	2
DEBUG-86 Commands .....	5
Monitor Commands .....	12
ISIS-II Error Messages .....	14
RUN Program Error Messages .....	15
DEBUG-86 Error Messages .....	15
Console Command Interface Errors .....	16
Hexadecimal-Decimal Conversion .....	17

## CONVENTIONS:

UPPERCASE—must be entered as shown

*lower case*—variable information

[ ]—indicate optional field

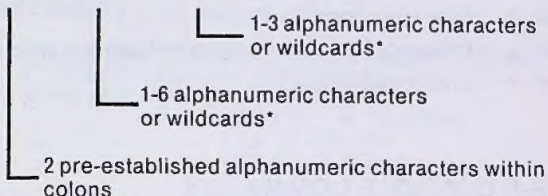
...—field may be repeated

{ }—one and only one entry must be selected

{ }...—at least one entry must be selected

## DEVICE FILENAME FORMAT

:device:filename.extension



\*wildcards:

An asterisk (\*) matches any sequence of characters.

A question mark (?) matches any single character.

## System Designated Device Names:

:F0: thru :F9: Directory on the disk in drive 0...9  
(shown as :Fn: in command syntax)

:TI: Teletypewriter keyboard

:TO: Teletypewriter printer

:TP: Teletypewriter punch

:TR: Teletypewriter reader

:VI: Video terminal keyboard

:VO: Video terminal screen

:HP: High-speed paper tape punch

:HR: High-speed paper tape reader

:LP: Line printer

:CI: Console input

:CO: Console output

:BB: Byte bucket

## CONTROL CHARACTERS

RUBOUT	Deletes preceding character
CNTL-C	Terminates 8086 program execution and enters RUN or ISIS-II
CNTL-D	Interrupts 8086 program execution and enters DEBUG-86
CNTL-E	In a SUBMIT file, switches the console input from the command sequence file to the initial system console
CNTL-P	Allows literal entry of control characters (including itself)
CNTL-Q	Resumes console display
CNTL-R	Redisplays current input line as modified
CNTL-S	Suspends console display
CNTL-X	Deletes all characters since last carriage return
CNTL-Z	Enters end-of-file

## ISIS-II CONSOLE COMMANDS

### Disk Maintenance

**FORMAT**—format a new disk and copy files

FORMAT :Fn:label [switches] <cr>

where *label* = name of disk

*switches* = A—copy all files

S—copy files with system attribute set

FROM n—identifies disk containing files needed for formatting

**IDISK**—format a new disk as a basic system or non-system disk

IDISK :Fn:label [switches] <cr>

where *label* = name of disk

*switches* = S—copy files needed for basic system disk

P—specifies single drive mode

FROM n—identifies disk containing files needed for formatting

**FIXMAP**—map bad sectors on a hard disk

FIXMAP *drive* <cr>

where *drive* = number of hard disk unit 0-1

Subcommands are:

MARK <i>disk-address</i> <cr>	Change the known state of a sector from good to bad.
FREE <i>disk-address</i> <cr>	Change the known state of a sector from bad to good.
LIST [ <i>filename</i> ] <cr>	List all known bad sectors.
COUNT <cr>	List the number of known bad sectors.
RECORD <cr>	Record changes specified by MARK and FREE.
QUIT <cr>	Exit to ISIS-II without recording changes.
EXIT <cr>	Record changes and exit to ISIS-II.

where *disk-address* is given as:

*track sector* [T]

*track* = 0-199    *sector* = 1-144

T = process 36 sectors at once

## File Control

ATTRIB—change and/or display the attributes of a disk file

ATTRIB [:Fn:]*filename* [*attriblist*] [Q] <cr>

where *attriblist* is: I0 or I1—invisible

W0 or W1—write protect

F0 or F1—format

S0 or S1—system

COPY—copy a file from one device to another

COPY [:Fn:]*infile* [... ] TO { [:Fn:]*outfile* } [*switches*] <cr>

where *switches* are: { :device: }

S = system—copy only files with S attribute

N = nonsystem—copy only files without F or S attribute

P = pause—single drive mode

Q = query—query before each copy

C = attribute—create *outfile* with same attributes as *infile*

B = brief—delete, then recreate *outfile* with new data

U = update—open *outfile* for update. Length changes only if *outfile* is extended

HDCOPY—copy the contents of one hard disk to another

HDCOPY { *indrive* TO *outdrive* } <cr>

BACKUP

*indrive* specifies the source disk. *outdrive* specifies the destination disk. The BACKUP option lets you backup a removable hard disk platter.

DELETE—remove references to a file from the directory

DELETE [:Fn:]*filename* [Q] [...][Q] [P] <cr>

where Q = query—query before each deletion

P = pause—single drive mode



**DIR**—output the names of and information about the files listed within the disk directory

**DIR** [FOR *filename*] [TO *listfile*] [*switches*] <cr>

where *switches* are:

- 0-9 —indicates drive number
- I —invisible—list invisible files
- F —fast—list only name.ext of files
- P —pause—single drive mode
- O —single column display
- Z —show number of sectors in use

**RENAME**—change the name of a disk file

**RENAME** [:Fn:]*oldname* TO [:Fn:] *newname* <cr>

Note—:Fn: must be the same in *oldname* and *newname*.

**VERS**—display ISIS utility version numbers

**VERS** [:Fn:] *filename* <cr>

where *filename* is the name of the ISIS file on :Fn:  
whose version number is to be displayed.

## 8080/8085 Program Execution

**filename**—execute the named program

*filename* [*parameters*] <cr>

where *filename* is the name of an 8080/8085 absolute object module to be executed.

*parameters* are parameters needed by *filename*

**DEBUG**—load an 8080/8085 program and give control to the Monitor

**DEBUG** [:Fn:] *filename* [*parameters*] <cr>

where *filename* is the name of the absolute object module to be debugged.

*parameters* are parameters needed by *filename*

**SUBMIT**—enter a file that contains commands to be executed

**SUBMIT** [:Fn:] *filename* [(parameter [...])9] <cr>

where *filename* is the name of the file containing the command sequence definition. If extension is omitted, SUBMIT looks for the default extension .CSD.

*parameter* specifies real values that replace formal parameters in the command sequence definition.

## 8086 Program Execution

**RUN**—activate the 8086 execution mode

**RUN** [[:Fn:] *filename* [*parameters*]]<cr>  
where *filename* is the name of an executable 8086 program. If you enter no extension, the system assumes a default extension of .86. If you enter an extension (or a period and no extension, as in MYPROG.), the default extension is not assumed.

*parameters* are parameters needed by *filename*.

**WORK**—change/display workfile default drive

[**RUN**] **WORK** [[:Fn:]]<cr>

where :Fn: specifies the drive n to be set as the default drive for temporary workfiles. Initial system default is :F1:. If :Fn: is not specified the current default is displayed.

**DATE**—change/display system date

[**RUN**] **DATE** [nn/nn/nn]<cr>

where nn = 00-99 specifying the date desired. If date is not specified, the last date entered is displayed.

**EXIT**—transfer control from RUN to ISIS-II

**EXIT**<cr>

## DEBUG-86 COMMANDS

### Utility Commands

**DEBUG**—Transfer Control to DEBUG-86

[[:Fn:]]**RUN** **DEBUG** [[:Fn:]*filename* [*parameters*]]<cr>

where *filename* is the name (including extension) of a program that is a valid absolute, PIC, or LTL 8086 object module. If an extension (or name plus period) is not specified, default extension of .86 is assumed.

*parameters* are ASCII characters (separated by commas or spaces) representing variable data required by your program.

**EXIT**—Exit DEBUG-86

**EXIT**<cr>

## LOAD—Load 8086 Object Code

LOAD [:Fn:]*filename* [ { NOSYMBOL  
NOLINE } ... ] <cr>

where *filename* is the complete name of a valid absolute, PIC or LTL 8086 object module. No default extension is assumed.

NOSYMBOL prevents program symbol table from being loaded.

NOLINE prevents program line number table from being loaded.

## Execution Commands

### GO—Execute 8086 Instructions

GO [FROM *addr*]  
{ [FOREVER]  
[TILL *break-addr* [OR *break-addr*]]  
[TILL *break-reg* [OR *break-reg*]] } <cr>

where FROM *addr* specifies the address of the first instruction to be executed. If it is omitted, the CS:IP address is used. Use the form *nnnn:nnnn*, as in 800:0 (leading zeros need not be entered).

*break-addr* is an integer expression entered as a pointer that references a 20-bit execution address.

*break-reg* is BR0 or BR1.

## GR Command

Display form:

GR<cr>

Change form:

GR =  
{ FOREVER  
TILL *break-addr* [OR *break-addr*]  
TILL *break-reg* [OR *break-reg*] } <cr>

where *break-addr* is an integer expression entered as a pointer that references a 20-bit execution address.



*break-reg* is BR0 or BR1 (or BR for both break-point registers).

## STEP—Execute a Single Instruction

STEP [FROM *address*] <cr>

where FROM *address* is the address where single step execution is to begin. If it is omitted, the CS:IP address is used. Use the form *nnnn:nnnn*, as in 800:0 (leading zeros need not be entered).

## Change Commands

### Change Register—Change Content of a Register

*register* = *change-exp* <cr>

where *register* is RAL, RAH, RBL, RBH, RCL, RCH, RDL, RDH, RAX, RBX, RCX, RDX, SP, BP, SI, DI, SS, CS, DS, ES, IP, RF, CFL, PFL, AFL, ZFL, SFL, TFL, IFL, DFL, or OFL.

*change-exp* is the new contents of *register*.

### Change Memory—Change Contents of Memory Locations

$$\text{memory-type } \text{addr} \left\{ \begin{array}{l} [\text{TO end-addr}] \\ [\text{LENGTH } n] \end{array} \right\} = \text{change-exp } [,...]19 \text{ <cr>}$$

where *memory-type* is BYTE, WORD, SINTEGER, INTEGER, or POINTER.

*addr* is a memory location entered as a pointer value containing a base and a displacement.

TO *end-addr* specifies the upper limit of a range of memory.

LENGTH *n* specifies a number of bytes, words, or pointers (depending on *memory-type*).

*change-exp* is the new contents of the specified memory location and is a pointer value if *memory-type* = pointer; otherwise, it is an integer value.

## Change Port—Change Contents of I/O Ports

*port-type* *addr*  $\left\{ \begin{array}{l} [\text{TO } \textit{end-addr}] \\ [\text{LENGTH } n] \end{array} \right\} =$   
*change-exp* [...]<sup>19</sup><cr>

where *port-type* is one of the following: PORT, WPORT.

*addr* is the address of an 8086 port and is an integer value between 0 and 65,535.

TO *end-addr* specifies the upper limit of a range of port addresses and is an integer value between 0 and 65,535.

LENGTH *n* specifies an integer value giving the number of port or word port addresses.

*change-exp* is the new contents of the specified port.

## Display Commands

### Display Register—Display Contents of 8086 Registers

$\left\{ \begin{array}{l} \textit{register} [, \dots]^{19} \\ \text{REGISTER} \\ \text{FLAG} \end{array} \right\} <\text{cr}>$

Up to 19 register keywords can be listed (see Change Memory Command.)

REGISTER displays all 16-bit 8086 registers.

FLAG displays all 1-bit status flags.

### Display Memory—Display 8086 Memory

*memory-type* *address*  $\left\{ \begin{array}{l} [\text{TO } \textit{end-address}] \\ [\text{LENGTH } n] \end{array} \right\} <\text{cr}>$

where *memory-type* is BYTE, WORD, SINTEGER, INTEGER, or POINTER.

*address* is a memory location entered as a pointer value containing a base and a displacement.

TO *end-address* specifies the upper limit of a range of memory.

LENGTH *n* specifies a number of bytes, words, or pointers (depending on *memory-type*).

## Display Memory—Display 8086 Memory in ASM Form

$$ASM\ address \left\{ \begin{array}{l} [TO\ end\ address] \\ [LENGTH\ n] \end{array} \right\} <cr>$$

where *memory-type* is BYTE, WORD, SINTEGER, INTEGER, or POINTER.

*address* is a memory location entered as a pointer value containing a base and a displacement.

TO *end-address* specifies the upper limit of a range of memory.

LENGTH *n* specifies a number of bytes, words, or pointers (depending on *memory-type*).

## Display Port—Display I/O Port Contents

$$port\text{-}type\ address \left\{ \begin{array}{l} [TO\ end\ address] \\ [LENGTH\ n] \end{array} \right\} <cr>$$

where *port-type* is either PORT or WPORT.

*address* is the address of an 8086 port and is an integer value between 0 and 65,535.

TO *end-address* specifies the upper limit of a range of port addresses and is an integer value between 0 and 65,535.

LENGTH *n* specifies an integer value giving the number of port or word port addresses.

## Display Boolean—Display Boolean Value

BOOL *expression* <cr>

where *expression* is evaluated to a boolean value. If the least significant bit of the result equals 1, the boolean value is TRUE; otherwise the boolean value is FALSE.

## Display Stack—Display User Stack Contents

STACK *expression* <cr>

where *expression* defines the number of words on the user stack to be displayed.

## EVALUATE—Display Integers in Five Bases

EVALUATE *expression* [SYMBOLICALLY] <cr>

where *expression* is an integer expression.

SYMBOLICALLY displays each numeric value output by the command as a symbol or a source statement, plus a remainder.

## Symbol Manipulation Commands

### Define Symbol—Enter New Symbol

DEFINE [*..module*].*symbol* = *change-exp* [OF *memory-type*] <cr>

where *module* is the name of an existing program module in which *symbol* is to be located.

*symbol* is a user-defined symbol to be entered into the symbol table.

*change-exp* is the address of statement labels or variables, or the value of a constant.

OF *memory-type* specifies any of the following: BYTE, WORD, SINTEGER, INTEGER, or POINTER.

### Display Symbols—Display One or More Symbols

{ SYMBOL  
[*..module*].*symbol* [*symbol*] ... } <cr>

where SYMBOL causes the entire DEBUG-86 symbol table to be displayed.

*symbol* is the name of an existing symbol.

## Display Lines—Display Statement Numbers

$$\left\{ \begin{array}{c} \text{LINE} \\ \text{[..module]\#statement-number} \end{array} \right\} \langle \text{cr} \rangle$$

where LINE displays all statement numbers in the current domain.

*statement-number* is the source statement number having a default decimal suffix.

## Display Modules—Display Module Names

MODULE<cr>

## Change Symbols—Change Value of a Symbol

*[..module].symbol[.symbol ...] ... =  
change-exp [OF memory-type]<cr>*

## Remove Symbols Command

$$\text{REMOVE} \left\{ \begin{array}{l} \text{[..module].symbol [..symbol...]}19 \text{ ,...} \\ \text{SYMBOL} \\ \text{MODULE ..module [..module]...} \end{array} \right\} \langle \text{cr} \rangle$$

where up to 19 modules and 19 symbols can be listed.

SYMBOL deletes entire current DEBUG-86 symbol table.

MODULE deletes all symbols and lines of the named module from the symbol and statement number tables.

## Set Domain Command

DOMAIN ..module <cr>

where DOMAIN establishes a default module for source statement number references.

*module* is the name of an existing program module.



## Compound Commands

### REPEAT Command

```
REPEAT<cr>  
[ command<cr>  
  WHILE boolean-expression<cr> ] ...  
  UNTIL boolean-expression<cr> ]  
END<cr>
```

### COUNT Command

```
COUNT arithmetic-expression<cr>  
[ command<cr>  
  WHILE boolean-expression<cr> ] ...  
  UNTIL boolean-expression<cr> ]  
END<cr>
```

### IF Command

```
IF boolean-expression [THEN]<cr>  
  [ command<cr> ] ...  
[ ORIF boolean-expression [THEN]<cr> ] ...  
  [ command<cr> ] ...  
[ ELSE<cr>  
  [ command<cr> ] ... ]  
END<cr>
```

## MONITOR COMMANDS

### Monitor I/O Configuration Command

#### A—Assign Command

*A logical-device = physical-device* <cr>

Possible values of logical and physical device are:

Logical Device	Physical Device
C or Console	T or TTY
	C or CRT
	B or BATCH
	1 (reserved)

R or Reader	T or TTY P or PTR 1 or 2 (reserved)
P or Punch	T or TTY P or PTP 1 or 2 (reserved)
L or List	T or TTY C or CRT L or LPT 1 (reserved)

## **Q—Query Command**

*Q <cr>*

## **Memory Control Commands**

### **D—Display Memory**

*D start-address, end-address <cr>*

### **F—Fill Memory**

*F start-address, end-address, constant <cr>*

### **M—Move Memory**

*M start-address, end-address, destination-address <cr>*

### **S—Substitute Memory**

*S address, [data-byte][,[data-byte]][...] <cr>*

## **Register Commands**

### **X—Register Command**

Display Form: *X <cr>*

Modify Form:

*X register, [data][,[data]][,...] <cr>*

## **Paper Tape I/O Commands**

### **R—Read**

*R bias <cr>*

### **W—Write**

*W start-address, end-address <cr>*

### **E—End of File**

*E entry-point <cr>*

### **N—Null**

*N <cr>*

## **Execute Command**

### **G—Execute Command**

*G [start-address][, breakpoint 1[, breakpoint 2]] <cr>*

## **Utility Command**

### **H—Hexadecimal add and subtract**

*H number 1, number 2 <cr>*

## ISIS-II ERROR MESSAGES

1. Fatal error. Too few buffers were allocated.
2. Illegal active file table number.
3. Fatal error. Active file table is full.
4. Incorrectly specified filename.
5. Unrecognized device name.
6. Attempt to write to input device.
7. Fatal error. The disk is full.
8. Attempt to read from output device.
9. Disk directory is full.
10. Pathname is not on same disk
11. File already exists.
12. File is already open.
13. No such file.
14. Write-protected file encountered.
15. Fatal error. ISIS overwrite.
16. Fatal error. Bad load format.
17. Not a disk file.
18. Illegal ISIS commands.
19. Attempted seek on non-disk file.
20. Attempted back seek too far.
21. Can't rescan.
22. Illegal access mode to open.
23. Missing filename.
24. Fatal error. Disk input/output hardware error. See note below.
25. Illegal echo file.
26. Illegal attribute identifier.
27. Illegal seek command.
28. Missing extension.
29. Fatal error. Premature EOF
30. Fatal error. Drive not ready.
31. Can't seek on write only file.
32. Can't delete open file.
33. Fatal error. Illegal system call parameter.
34. Fatal error. Invalid return switch in a LOAD system call.
35. Seek past EOF.

When error 24 occurs, an additional message is displayed:

```
STATUS=00nn  
D=x T=yyy S=zzz
```

where x represents the drive number, yyy the track address, zzz the sector address, and where nn has the following meanings:

For flexible disks:

- |    |                      |
|----|----------------------|
| 01 | Deleted record       |
| 02 | Data field CRC error |
| 03 | Invalid address mark |

- 04 Seek error
- 08 Address error
- 0A ID field CRC error
- 0E No address mark
- 0F Incorrect data address mark
- 10 Data overrun or data underrun
- 20 Attempt to write on Write Protect
- 40 Drive has indicated a Write error
- 80 Drive not ready

For hard disks:

- 01 ID field miscompare
- 02 Data field CRC error
- 04 Seek error
- 08 Bad sector address
- 0A ID field CRC error
- 0B Protocol violations
- 0C Bad track address
- 0E No ID address mark or sector not found
- 0F Bad data field address mark
- 10 Format error
- 20 Attempt to write on write-protected drive
- 40 Drive has indicated a write error
- 80 Drive not ready

## RUN PROGRAM ERROR MESSAGES

- 101. HARDWARE NOT RESPONDING (fatal error)
- 102. INVALID SYNTAX
- 103. COMMAND LINE TOO LONG
- 104. INSUFFICIENT MEMORY TO LOAD
- 105. MISMATCHED SOFTWARE/FIRMWARE
- 106. ERROR 106 USER PC mmmm
- 107. ILLEGAL LOAD ADDRESS
- 108. INVALID OBJECT FILE
- 117. UNRESOLVED SYMBOLS (warning)
- 118. RAM FAILURE (warning)
- 119. ROM CHECKSUM ERROR (warning)

## DEBUG-86 ERROR MESSAGES

- 120. Syntax error
- 121. Invalid token
- 122. No such line
- 123. Inappropriate number
- 124. Partition bounds error
- 125. Symbol already exists
- 126. Symbol does not exist
- 127. Memory failure

- 133. Null string error
- 134. Memory overflow
- 135. Stack overflow
- 136. Command too complex
- 137. Module does not exist
- 139. Excessive data
- 141. Unsuitable execute file
- 142. Line too long
- 143. Too many partitions
- 147. Pointer value required
- 148. Integer value required
- 149. Differing bases

## CONSOLE COMMAND INTERFACE ERRORS

- 201. Unrecognized switch
- 202. Unrecognized delimiter
- 203. Invalid syntax
- 206. Illegal disk label
- 208. Checksum error
- 209. Relo file sequence error
- 210. Insufficient memory
- 211. Record too long
- 212. Illegal relo type
- 213. Fixup bounds error
- 214. Illegal SUBMIT parameter
- 215. Argument too long
- 216. Too many parameters
- 217. Object record too short
- 218. Illegal record format
- 219. Phase error
- 220. No EOF record in object module file
- 221. Segment overflow during LINK operation
- 222. Unrecognized record in object module file
- 223. Fixup record pointer is incorrect
- 224. Illegal record sequence in object module file in LINK
- 225. Illegal module name specified
- 226. Module name exceeds 31 characters
- 227. Command syntax requires left parenthesis
- 228. Command syntax requires right parenthesis
- 229. Unrecognized control specified in command
- 230. Duplicate symbol found
- 231. File already exists
- 232. Unrecognized command
- 233. Command syntax requires a TO clause
- 234. Filename illegally duplicated in command
- 235. File specified in command is not a library file
- 236. More than 249 common segments in input files
- 237. Specified common segment not found in object file
- 238. Illegal stack content record in object file
- 239. No module header in input object file
- 240. Program exceeds 64K bytes



# HEXADECIMAL-DECIMAL CONVERSION

BYTE				BYTE			
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4,096	1	256	1	16	1	1
2	8,192	2	512	2	32	2	2
3	12,288	3	768	3	48	3	3
4	16,384	4	1,024	4	64	4	4
5	20,480	5	1,280	5	80	5	5
6	24,576	6	1,536	6	96	6	6
7	28,672	7	1,792	7	112	7	7
8	32,768	8	2,048	8	128	8	8
9	36,864	9	2,304	9	144	9	9
A	40,960	A	2,560	A	160	A	10
B	45,056	B	2,816	B	176	B	11
C	49,152	C	3,072	C	192	C	12
D	53,248	D	3,328	D	208	D	13
E	57,344	E	3,584	E	224	E	14
F	61,440	F	3,840	F	240	F	15



**3065 Bowers Avenue, Santa Clara, California 95051  
(408) 987-8080**

Printed in U.S.A.  
A 461/1081/12K DD